All-Instances Restricted Chase Termination for Linear TGDs

KI - Künstliche Intelligenz (2020) 34:465–473
https://doi.org/10.1007/s13218-020-00690-7

TECHNICAL CONTRIBUTION

All-Instances Restricted Chase Termination for Linear TGDs

Tomasz Gogacz[1] · Jerzy Marcinkowski[2] · Andreas Pieris[3]

Received: 6 March 2020 / Accepted: 2 September 2020 / Published online: 26 September 2020

Abstract
The chase procedure is a fundamental algorithmic tool in database theory with a variety of applications. A key problem concerning the chase procedure is all-instances chase termination: for a given set of tuple-generating dependencies (TGDs), is it the case that the chase terminates for every input database? In view of the fact that this problem is, in general, undecidable, it is natural to ask whether well-behaved classes of TGDs, introduced in different contexts, ensure decidability. It has been recently shown that the problem is decidable for the restricted (a.k.a. standard) version of the chase, and linear TGDs, a prominent class of TGDs that has been introduced in the context of ontological query answering, under the assumption that only one atom appears in TGD-heads. We provide an alternative proof for this result based on Monadic Second-Order Logic, which we believe is simpler that the ones obtained from the literature.

1 Introduction

The *chase procedure* (or simply chase) is a fundamental algorithmic tool that has been successfully applied to several database problems such as computing data exchange solutions [14], query answering under constraints [9], containment of queries under constraints [1], and checking logical implication of constraints [5, 22], to name a few. It accepts as an input a database D and a set $\mathcal{T}$ of constraints—which, for this work, are tuple-generating dependencies (TGDs) of the form $\forall\bar{x}\forall\bar{y}(\phi(\bar{x},\bar{y}) \rightarrow \exists\bar{z}\,\psi(\bar{x},\bar{z}))$ with ϕ and ψ being conjunctions of atoms – and, if it terminates, its result is a finite instance $D_{\mathcal{T}}$ that is a *universal model* of D and $\mathcal{T}$, i.e., is a model that can be homomorphically mapped into every other model of D and $\mathcal{T}$. This is the reason for the ubiquity of the chase in database theory. Indeed, many key database problems can be solved by simply exhibiting a universal model. And this is not only in theory. Despite the fact that the instance constructed by the chase can be very large, efficient implementations of the chase procedure have been successfully applied during the last few years in many different contexts [6, 20, 25, 26].

Given a database D and a set $\mathcal{T}$ of TGDs, roughly speaking, the chase adds new atoms to D (possibly involving null values that act as witnesses for the existentially quantified variables) until the final result satisfies $\mathcal{T}$. Here is a simple example of how the chase procedure works.

Example 1 Given the database $D = \{R(c)\}$, and the TGDs

$$\forall x(R(x) \rightarrow \exists y\, P(x,y)) \quad \text{and} \quad \forall x\forall y(P(x,y) \rightarrow R(y)),$$

the database atom *triggers* the first TGD, and the chase adds in D the atom $P(c,\bot_1)$, which in turn triggers the second TGD and $R(\bot_1)$ is added, where $\bot_1$ is a labeled null representing some unknown value.

However, the atom $R(\bot_1)$ triggers again the first TGD, and the chase adds the atom $P(\bot_1,\bot_2)$, which triggers again the second TGD.

The result of the chase is eventually the infinite instance

$$\{R(c), P(c,\bot_1)\} \cup \bigcup_{i>0} \{R(\bot_i), P(\bot_i,\bot_{i+1})\},$$

where $\bot_1, \bot_2, \bot_3, \cdots$ are labeled null values. □

✉ Andreas Pieris
apieris@inf.ed.ac.uk

Tomasz Gogacz
t.gogacz@mimuw.edu.pl

Jerzy Marcinkowski
jma@cs.uni.wroc.pl

[1] Institute of Informatics, University of Warsaw, Warsaw, Poland

[2] Institute of Computer Science, University of Wroclaw, Wroclaw, Poland

[3] School of Informatics, University of Edinburgh, Edinburgh, UK

1.1 The Challenge of Non-termination

As said above, there are nowadays efficient implementations of the chase that allows us to solve central database problems by adopting a materialization-based approach [6, 20, 25, 26]. But, of course, for this to be feasible in practice we need a guarantee that the chase terminates, which, as shown by Example 1, it is not always the case. This fact motivated a long line of research on identifying fragments of TGDs that ensure the termination of the chase procedure, for every input database. A prime example is the class of *weakly-acyclic* TGDs [14], the standard language for data exchange purposes, that guarantees the termination of the semi-oblivious and restricted (a.k.a. standard) chase. A similar formalism, called *constraints with stratified-witness*, has been proposed in [13]. Inspired by weak-acyclicity, the notion of *rich-acyclicity* has been proposed in [19], which guarantees the termination of the oblivious chase. Many other sufficient conditions can be found in the literature; see, for example, [12, 13, 17, 18, 23, 24]. At this point, let us note that the restricted chase applies a TGD only if it is necessary, i.e., only if the TGD is violated, while the (semi-)oblivious chase applies TGDs whenever the body is satisfied, without checking whether the head is satisfied.

With so much effort spent on identifying sufficient conditions for the termination of the chase procedure, the question that comes up is whether a sufficient condition that is also *necessary* exists. In other words, given a set $\mathcal{T}$ of TGDs, is it possible to decide whether, for *every* database D, the chase on D and $\mathcal{T}$ terminates? This has been studied in [15], and has been shown that the answer is negative, no matter which version of the chase we consider, namely the oblivious, semi-oblivious and restricted chase.

The undecidability proof in [15] relies on a sophisticated set of TGDs that goes beyond existing well-behaved classes of TGDs that enjoy certain syntactic properties, which in turn ensure useful model-theoretic properties. Such well-behaved classes of TGDs have been proposed in the context of ontological reasoning. The two main paradigms that led to robust TGD-based languages are *guardedness* [2, 9, 10] and *stickiness* [11]. A TGD is guarded if the left-hand side of the implication, known as the body of the TGD, has an atom that contains (or "guards") all the universally quantified variables. If a TGD has only one body-atom, which is trivially a guard, then is called *linear*; the class of linear TGDs is actually the main concern of the present work. On the other hand, sticky sets of TGDs are inherently unguarded. The key idea underlying stickiness can be described as follows: variables that appear more than once in the body of a TGD should be inductively propagated (or "stick") to every atom in the right-hand side (the head) of the TGD. Observe that the set of TGDs given in Example 1 is both guarded (actually linear) and sticky; notice that stickiness holds trivially since every body-variable occurs only once.

The fact that the set of TGDs given in the undecidability proof of [15] is far from being guarded (and therefore linear) or sticky raised the following question: is the chase termination problem, as described above, decidable for linear, guarded or sticky sets of TGDs? This question is rather well-understood for the (semi-)oblivious chase. In the case of linear TGDs, the problem is PSPACE-complete, and becomes 2EXPTIME-complete for guarded TGDs [7]. The sticky case has been recently addressed in [8], where it is shown that the problem is PSPACE-complete. On the other hand, when it comes to the more subtle case of the restricted chase, the problem has been studied only for single-head TGDs, i.e., TGDs with only one atom in the head, while the general case remains open. It has been recently shown that the problem is decidable for single-head guarded (and hence linear) and sticky TGDs [16]. The same result for single-head linear TGDs has been independently shown in [21].

1.2 Our Main Objective

In this work, we concentrate on single-head linear TGDs, and provide an alternative proof for the decidability of the restricted chase termination problem, which we believe is simpler than the ones obtained from the literature [16, 21]. More precisely, we focus on the following problem: given a set $\mathcal{T}$ of single-head linear TGDs, is it the case that for *every* database D, *every* restricted chase derivation of D w.r.t. $\mathcal{T}$ is finite? Note that, in general, it might be the case that some derivations are finite and some others are not, depending on the order that the TGDs are being triggered, which is not the case for the (semi-)oblivious chase. The reason for this non-deterministic behavior is the fact that, as explained above, the restricted chase applies a TGD only if it is necessary, whereas the (semi-)oblivious chase applies TGDs whenever the body is satisfied, without checking whether the head is satisfied, which ensures a deterministic behavior.

As mentioned above, the decidability of this problem has been recently shown independently in [16, 21]. In fact, [16] shows that the problem is decidable for the class of single-head guarded TGDs, which generalizes single-head linear TGDs. This is done via a reduction to the satisfiability problem of Monadic Second-Order Logic (MSOL) over infinite trees of bounded degree. On the other hand, [21] concentrates on the class of single-head linear TGDs, and the decidability of the restricted chase termination problem is shown by relying on derivation trees, a notion that was originally introduced in the context of ontological query answering [3]. Let us also say that the proof given in [16] for single-head sticky TGDs, which is via a reduction to the

emptiness problem of deterministic Büchi automata, can be converted into a proof for single-head linear TGDs.

Although several different proofs for the decidability of the restricted chase termination problem for single-head linear TGDs can be obtained from the literature, we strongly believe that a proof based on MSOL that directly exploits the linearity of the TGDs is the natural way to go. This will provide a neat solution to the problem in question via standard means, which is simpler than the existing ones. The main objective of this work is to provide such a proof.

2 Preliminaries

We consider the disjoint countably infinite sets $\mathbf{C}$, $\mathbf{N}$, and $\mathbf{V}$ of *constants*, *(labeled) nulls*, and *variables*, respectively. We refer to constants, nulls and variables as *terms*. For $n > 0$, we may write $[n]$ for the set $\{1, \ldots, n\}$.

Relational Databases. A *schema* $\mathbf{S}$ is a finite set of relation symbols (or predicates) with associated arity. We write R/n to denote that R has arity $n > 0$; we may also write $\mathsf{ar}(R)$ for n. A position of $\mathbf{S}$ is a pair (R, i), where $R/n \in \mathbf{S}$ and $i \in [n]$, that identifies the i-th argument of R. An *atom* over $\mathbf{S}$ is an expression of the form $R(\bar{t})$, where $R/n \in \mathbf{S}$ and $\bar{t}$ is an n-tuple of terms. A *fact* is an atom whose arguments consist only of constants. We write $R(\bar{t})[i]$ for the term of $R(\bar{t})$ at position (R, i), i.e., the i-th element of $\bar{t}$. An *instance* over $\mathbf{S}$ is a (possibly infinite) set of atoms over $\mathbf{S}$ that contain constants and nulls, while a *database* over $\mathbf{S}$ is a finite set of facts over $\mathbf{S}$. The *active domain* of an instance I, denoted $\mathsf{dom}(I)$, is the set of all terms in I.

Substitutions and Homomorphisms. A *substitution* from a set of terms T to a set of terms T' is a function $h : T \to T'$ defined as follows: $\emptyset$ is a substitution, and if h is a substitution, then $h \cup \{t \mapsto t'\}$, where $t \in T$ and $t' \in T'$, is a substitution. The restriction of h to $S \subseteq T$, denoted $h_{|S}$, is the substitution $\{t \mapsto h(t) : t \in S\}$. A *homomorphism* from a set of atoms A to a set of atoms B is a substitution h from the terms occurring in A to the terms occurring in B such that (i) $t \in \mathbf{C}$ implies $h(t) = t$, and (ii) $R(t_1, \ldots, t_n) \in A$ implies $h(R(t_1, \ldots, t_n)) = R(h(t_1), \ldots, h(t_n)) \in B$.

Single-Head Tuple-Generating Dependencies. A *single-head tuple-generating dependency* σ is a constant-free first-order sentence of the form

$$\forall \bar{x} \forall \bar{y} (\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, R(\bar{x}, \bar{z})),$$

where $\bar{x}, \bar{y}, \bar{z}$ are tuples of variables of $\mathbf{V}$, $\phi(\bar{x}, \bar{y})$ is a conjunction of atoms, and $R(\bar{x}, \bar{z})$ is a single atom. For brevity, we write σ as $\phi(\bar{x}, \bar{y}) \rightarrow \exists \bar{z}\, R(\bar{x}, \bar{z})$, and use comma instead of $\wedge$ for joining atoms. We refer to $\phi(\bar{x}, \bar{y})$ and $R(\bar{x}, \bar{z})$ as the *body* and *head* of σ, denoted $\mathsf{body}(\sigma)$ and $\mathsf{head}(\sigma)$, respectively. Henceforth, we simply say tuple-generating dependency (TGD) instead of single-head TGD. The *frontier* of the TGD σ, denoted $\mathsf{fr}(\sigma)$, is the set of variables $\bar{x}$, i.e., the variables that appear both in the body and in the head of σ. Note that, by abuse of notation, we sometimes treat a tuple of variables as a set of variables. The schema of a set $\mathcal{T}$ of TGDs, denoted $\mathsf{sch}(\mathcal{T})$, is the set of predicates occurring in $\mathcal{T}$, and we write $\mathsf{ar}(\mathcal{T})$ for the maximum arity over all those predicates. An instance I satisfies a TGD σ, written $I \models \sigma$, if the following holds: whenever there is a homomorphism h such that $h(\mathsf{body}(\sigma)) \subseteq I$, then there is $h' \supseteq h_{|\mathsf{fr}(\sigma)}$ such that $h'(\mathsf{head}(\sigma)) \in I$. By abuse of notation, we may treat a conjunction of atoms as a set. The instance I satisfies a set $\mathcal{T}$ of TGDs, written $I \models \mathcal{T}$, if $I \models \sigma$ for each $\sigma \in \mathcal{T}$.

Linearity. A TGD σ is called *linear* if $\mathsf{body}(\sigma)$ consists of a single atom [10]. The class of linear TGDs, denoted $\mathbb{L}$, is the family of all possible finite sets of linear TGDs.

3 The Restricted Chase Procedure

The chase procedure accepts as input a database D and a set $\mathcal{T}$ of TGDs, and constructs an instance that contains D and satisfies $\mathcal{T}$. Central notions in this context are the notion of trigger, and the notion of trigger application.

Definition 1 (Chase Trigger) A *trigger* for a set $\mathcal{T}$ of TGDs on an instance I is a pair (σ, h), where $\sigma \in \mathcal{T}$ and h is a homomorphism from $\mathsf{body}(\sigma)$ to I. We call (σ, h) *active* if there is no $h' \supseteq h_{|\mathsf{fr}(\sigma)}$ such that $h'(\mathsf{head}(\sigma)) \in I$. We denote by $\mathsf{result}(\sigma, h)$ the atom $\nu(\mathsf{head}(\sigma))$, where ν is a mapping from the variables of $\mathsf{head}(\sigma)$ to $\mathbf{N}$ defined as

$$\nu(x) = \begin{cases} h(x) & \text{if } x \in \mathsf{fr}(\sigma), \\ \perp^{x}_{\sigma,h} & \text{otherwise.} \end{cases}$$

An *application* of (σ, h) to I returns the instance

$$J = I \cup \{\mathsf{result}(\sigma, h)\},$$

and such an application is denoted as $I\langle \sigma, h \rangle J$. □

In the definition of $\mathsf{result}(\sigma, h)$, each existentially quantified variable x occurring in $\mathsf{head}(\sigma)$ is mapped by ν to a "fresh" null value of $\mathbf{N}$ whose name is uniquely determined by the trigger (σ, h) and x itself. Thus, given a trigger (σ, h), we can unambiguously write down the atom $\mathsf{result}(\sigma, h)$.

The main idea of the restricted chase is, starting from a database D, to apply active triggers for the given set $\mathcal{T}$ of TGDs on the instance constructed so far, and keep doing this until a fixpoint is reached. This is formalized as follows. Consider a database D and a set $\mathcal{T}$ of TGDs. We distinguish the two cases where the chase is terminating or not:

- A sequence $(I_i)_{0\leq i\leq n}$ of instances, with $D = I_0$ and $n \geq 0$, is a *restricted chase derivation* of D w.r.t. $\mathcal{T}$ if: for $0 \leq i < n$, there is an active trigger (σ, h) for $\mathcal{T}$ on I_i with $I_i\langle\sigma, h\rangle I_{i+1}$, and no active trigger for $\mathcal{T}$ on I_n.
- A sequence $(I_i)_{i\geq 0}$ of instances, with $D = I_0$, is a *restricted chase derivation* of D w.r.t. $\mathcal{T}$ if, for $i \geq 0$, there exists an active trigger (σ, h) for $\mathcal{T}$ on I_i such that $I_i\langle\sigma, h\rangle I_{i+1}$. Moreover, $(I_i)_{i\geq 0}$ is called *fair* if, for each $i \geq 0$, and every active trigger (σ, h) for $\mathcal{T}$ on I_i, there exists $j > i$ such that (σ, h) is not active for $\mathcal{T}$ on I_j. In a fair chase derivation all the active triggers will eventually be deactivated, which is not true for unfair ones.

A restricted chase derivation is called *valid* if it is finite, or infinite and fair. Infinite but unfair restricted chase derivations are not valid since they do not serve the main purpose of the chase procedure, i.e., build an instance that satisfies the given set of TGDs. Since we deal only with the restricted chase, in the rest of the paper we may simply say chase derivation meaning restricted chase derivation.

3.1 Chase Termination Problem

It is well-known that due to the existentially quantified variables, a valid chase derivation may be infinite. This is true even for very simple settings: it is easy to verify that the only chase derivation of $D = \{R(a, b)\}$ w.r.t. the set consisting of the single TGD $R(x, y) \rightarrow \exists z\, R(y, z)$ is infinite. The key question is, given a set $\mathcal{T}$ of TGDs, can we check whether, for every database D, every valid chase derivation of D w.r.t. $\mathcal{T}$ is finite? Before formalizing this problem, let us recall a central class of TGDs:

$$\mathbb{CT}^{\mathsf{res}}_{\forall\forall} = \left\{ \mathcal{T} : \begin{array}{l} \text{for every database } D, \\ \text{every valid restricted chase derivation} \\ \text{of } D \text{ w.r.t. } \mathcal{T} \text{is finite.} \end{array} \right\}$$

The superscript res in $\mathbb{CT}^{\mathsf{res}}_{\forall\forall}$ indicates that we concentrate on restricted chase derivations, while the subscript $\forall\forall$ indicates that we consider *every* database, and *every* valid chase derivation. The main problem tackled in this work is defined as follows, where $\mathbb{C}$ is a class of TGDs:

PROBLEM :	$\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{C})$
INPUT :	A set $\mathcal{T} \in \mathbb{C}$ of TGDs.
QUESTION :	Is it the case that $\mathcal{T} \in \mathbb{CT}^{\mathsf{res}}_{\forall\forall}$?

The above decision problem is, in general, undecidable. In fact, assuming that $\mathbb{TGD}$ is the class of arbitrary (single-head) TGDs, we have the following undecidability result:

Theorem 1 $\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{TGD})$ *is undecidable, even if we focus on schemas with binary predicates.*

Note that the undecidability of $\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{TGD})$ has been originally shown in [15] for schemas with binary and ternary predicates. The undecidability for schemas with binary predicates has been recently shown in [4] by adapting the proof of [15]. On the other hand, when it comes to the class of linear TGDs, we know that the above problem is decidable:

Theorem 2 $\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{L})$ *is decidable in elementary time.*

The above result has been recently shown independently in [16, 21]. In fact, [16] shows that the problem is decidable for the class of guarded TGDs, which generalizes linear TGDs. This is done via a reduction to the satisfiability problem of Monadic Second-Order Logic (MSOL) over infinite trees of bounded degree. On the other hand, [21] concentrates on the class of linear TGDs, and the decidability of the chase termination problem is shown by relying on derivation trees, a notion that was originally introduced in the context of ontological query answering [3]. The goal of the present work is to provide an alternative proof for Theorem 2 that relies on standard means, and is simpler than the ones obtained from [16, 21]. This is done by exploiting MSOL.

3.2 Dealing With Fairness

As one might expect, to prove the decidability of $\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{L})$, we focus on its complement and show that, for a set $\mathcal{T}$ of linear TGDs, we can decide whether there exists a database D such that there is a fair infinite chase derivation of D w.r.t. $\mathcal{T}$. This is precisely how Theorem 2 is shown in [16, 21]. However, as it has been already observed in [16, 21], the difficulty is to ensure fairness. Interestingly, we know the following:

Theorem 3 *Consider a database D, and a set $\mathcal{T}$ of single-head linear TGDs. If there exists an infinite restricted chase derivation of D w.r.t. $\mathcal{T}$, then there exists a fair one.*

The above has been independently shown in [16, 21]. Notice, however, that the proof of [21] applies only to linear TGDs, while [16] shows, via a more sophisticated proof, that the above holds for arbitrary single-head (not necessarily linear) TGDs. At this point, let us stress that Theorem 3 does *not* hold for TGDs that can have a conjunction of atoms in the head. This is illustrated by the following example:

Example 2 Consider the set $\mathcal{T}$ of TGDs consisting of

$R(x, y, y) \rightarrow \exists z\, R(x, z, y), R(z, y, y)$

$R(x, y, z) \rightarrow R(z, z, z)$.

There is an infinite restricted chase derivation of $\{R(a, b, b)\}$ w.r.t. $\mathcal{T}$; apply only the first TGD. However, every valid chase derivation of $\{R(a, b, b)\}$ w.r.t. $\mathcal{T}$ is finite.[1] □

The above discussion reveals the subtlety of the restricted chase, and explains why Theorem 2 is stated only for single-head TGDs. The decidability status of $\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{L}^{\wedge})$, where $\mathbb{L}^{\wedge}$ is the class of arbitary linear TGDs, where the head can be a conjunction of atoms, remains an open problem.

From Theorem 3, we get the following useful corollary:

Corollary 1 *Let $\mathcal{T} \in \mathbb{L}$. The following are equivalent*:

1. $\mathcal{T} \notin \mathbb{CT}^{\mathsf{res}}_{\forall\forall}$.
2. *There is a database D such that there exists an infinite chase derivation of D w.r.t. $\mathcal{T}$.*

Therefore, the complement of $\mathsf{CT}^{\mathsf{res}}_{\forall\forall}(\mathbb{L})$ boils down to the problem of checking whether there is a database D such that there exists an infinite chase derivation δ of D w.r.t. the given set $\mathcal{T} \in \mathbb{L}$, without having to ensure that δ is fair.

3.3 Plan of Attack

Our proof for Theorem 2 consists of two main steps:

1. We first establish, by relying on Corollary 1, that for a set $\mathcal{T} \in \mathbb{L}$, $\mathcal{T} \notin \mathbb{CT}^{\mathsf{res}}_{\forall\forall}$ iff there exists a so-called chase path for $\mathcal{T}$, which essentially encodes a path-like infinite chase derivation of a singleton database w.r.t. $\mathcal{T}$.
2. We then show that chase paths are MSOL-definable, i.e., we can devise an MSOL sentence $\Phi_{\mathcal{T}}$ over infinite paths that is satisfiable iff a chase path for $\mathcal{T}$ exists.

The rest of the paper is devoted to giving further details concerning the above two steps.

4 Non-termination via Chase Paths

We start by introducing the notion of chase path. Given a trigger (σ, h) for $\mathcal{T}$ on some instance I, and an atom α, we say that α *stops* $\mathsf{result}(\sigma, h)$, written $\alpha \blacktriangleright \mathsf{result}(\sigma, h)$, if there exists a homomorphism h' such that

1. $h'(\mathsf{result}(\sigma, h)) = \alpha$, and
2. $h'(h(x)) = h(x)$ for each $x \in \mathsf{fr}(\sigma)$.

Roughly speaking, $\alpha \blacktriangleright \mathsf{result}(\sigma, h)$ means that in the presence of α the atom $\mathsf{result}(\sigma, h)$ is superfluous in the sense that the trigger (σ, h) for $\mathcal{T}$ on an instance that contains α is not active due to the presence of α. This is summarized in the following fact, which is easy to verify:

Fact 1 Let $\mathcal{T} \in \mathbb{L}$, and (σ, h) be a trigger for $\mathcal{T}$ on some instance I over $\mathsf{sch}(\mathcal{T})$. The following are equivalent:

1. (σ, h) is active.
2. There is no $\alpha \in I$ such that $\alpha \blacktriangleright \mathsf{result}(\sigma, h)$.

We are now ready to introduce the notion of chase path.

Definition 2 (Chase Path) Let $\mathcal{T} \in \mathbb{L}$. A *chase path for $\mathcal{T}$* is an infinite sequence $(\alpha_i)_{i \geq 0}$ of atoms over $\mathsf{sch}(\mathcal{T})$, which contain constants and nulls, such that:

1. α_0 is a fact, i.e., it contains only constants.
2. For $i > 0$, there is a trigger (σ, h) for $\mathcal{T}$ on $\{\alpha_{i-1}\}$ with
   (a) $\alpha_i = \mathsf{result}(\sigma, h)$, and
   (b) there is no $0 \leq j < i$ such that $\alpha_j \blacktriangleright \mathsf{result}(\sigma, h)$.

□

Our goal is to show Lemma 1 given below. But first, we need to recall a useful notion known as the chase relation [11], which essentially describes how the atoms generated during the chase depend on each other. Consider a chase derivation $\delta = (I_i)_{i \geq 0}$ of a database D w.r.t. a set $\mathcal{T}$ of TGDs such that, for $i \geq 0$, $I_i \langle \sigma_i, h_i \rangle I_{i+1}$, i.e., $I_{i+1} = I_i \cup \{\mathsf{result}(\sigma_i, h_i)\}$. The *chase relation* of δ, denoted $\prec_\delta$, is a binary relation over $\bigcup_{i \geq 0} I_i$ such that $\alpha \prec_\delta \beta$ iff there is $i \geq 0$ with $\alpha \in h_i(\mathsf{body}(\sigma_i))$ and $I_{i+1} \setminus I_i = \{\beta\}$.

Lemma 1 *Let $\mathcal{T} \in \mathbb{L}$. The following are equivalent*:

1. $\mathcal{T} \notin \mathbb{CT}^{\mathsf{res}}_{\forall\forall}$.
2. *There exists a chase path for $\mathcal{T}$.*

Proof By Corollary 1, it suffices to show that the following statements are equivalent:

1. There exists a database D such that there is an infinite chase derivation of D w.r.t. $\mathcal{T}$.
2. There exists a chase path for $\mathcal{T}$.

We first show that (1) ⇒ (2). Observe that it suffices to show that (1) implies that there is a fact α such that there is an

[1] Note that [21] provides an example that refutes Theorem 3 even if we focus on binary predicates.

infinite chase derivation $\delta_\alpha = (\{\alpha_0, \cdots, \alpha_i\})_{i\geq 0}$, where $\alpha = \alpha_0$, of $\{\alpha_0\}$ w.r.t. $\mathcal{T}$ that enjoys the following property:

($\star$) for each $i \geq 0$, $\alpha_i \prec_{\delta_\alpha} \alpha_{i+1}$.

Indeed, in this case, for each $i > 0$, there is a trigger (σ, h) for $\mathcal{T}$ on $\{\alpha_{i-1}\}$ such that (a) $\alpha_i = \mathsf{result}(\sigma, h)$, and (b) (σ, h) is an active trigger for $\mathcal{T}$ on $\{\alpha_0, \ldots, \alpha_{i-1}\}$, and thus, by Fact 1, there is no $0 \leq j < i$ such that $\alpha_j \blacktriangleright \mathsf{result}(\sigma, h)$. Therefore, $(\alpha_i)_{i\geq 0}$ is a chase path for $\mathcal{T}$, which in turn implies that (1) $\Rightarrow$ (2), as needed. It remains to show that (1) implies the existence of δ_α as above.

By hypothesis, there is an infinite chase derivation $\delta = (I_i)_{i\geq 0}$ of D w.r.t. $\mathcal{T}$. Since the TGDs of $\mathcal{T}$ are linear, that is, they have only one atom in the body, it is easy to see that the chase relation $\prec_\delta$ is essentially a forest with its roots being the atoms of D, i.e., for each $\beta \in \bigcup_{i\geq 0} I_i$, $\gamma \prec_\delta \beta$ and $\gamma' \prec_\delta \beta$ implies $\gamma = \gamma'$. This allows us, for each $\alpha \in D$, to extract from δ a (finite or infinite) chase derivation δ'_α of $\{\alpha\}$ w.r.t. $\mathcal{T}$. Moreover, since δ is infinite, there exists at least one $\alpha \in D$ such that δ'_α is infinite. Therefore, δ'_α is an infinite chase derivation of $\{\alpha\}$ w.r.t. $\mathcal{T}$. However, δ'_α does not necessarily enjoy ($\star$). Note that $\prec_{\delta'_\alpha}$ is essentially a tree T_α rooted at α. It is not difficult to verify that the out-degree of T_α is bounded by $|\mathcal{T}| \cdot \mathsf{ar}(\mathcal{T})^{\mathsf{ar}(\mathcal{T})}$, and therefore finite since $\mathcal{T}$ is finite. Since T_α is infinite, by König's Lemma, we get that in T_α there is an infinite path $\alpha_0, \alpha_1, \alpha_2, \ldots$, where $\alpha_0 = \alpha$.[2] Clearly, $\alpha_i \prec_{\delta'_\alpha} \alpha_{i+1}$, for each $i \geq 0$. Therefore, $\delta_\alpha = (\{\alpha_0, \ldots, \alpha_i\})_{i\geq 0}$ is an infinite chase derivation of $\{\alpha\}$ w.r.t. $\mathcal{T}$ that enjoys ($\star$), and the claim follows.

Let us now show that (2) $\Rightarrow$ (1). By hypothesis, there is a chase path $(\alpha_i)_{i\geq 0}$ for $\mathcal{T}$. Let $\delta = (I_i)_{i\geq 0}$, where $I_i = \{\alpha_0, \ldots, \alpha_i\}$. It is clear that, for each $i > 0$, there is a trigger (σ, h) for $\mathcal{T}$ on I_{i-1} such that $I_{i-1}\langle\sigma, h\rangle I_i$. Moreover, there is no $\alpha \in \{\alpha_0, \ldots, \alpha_{i-1}\}$ such that $\alpha \blacktriangleright \mathsf{result}(\sigma, h)$; thus, by Fact 1, (σ, h) is active. This implies that δ is an infinite chase derivation of the singleton database $\{\alpha_0\}$ w.r.t. $\mathcal{T}$. □

5 Chase Paths are MSOL-definable

We proceed to show that the existence of a chase path for a set $\mathcal{T} \in \mathbb{L}$ can be checked via an MSOL sentence. In other words, we are going to argue that there exists an MSOL sentence $\Phi_\mathcal{T}$ such that $\Phi_\mathcal{T}$ is satisfiable over infinite paths iff there is a chase path for $\mathcal{T}$. Instead of giving the rather long and tedious sentence $\Phi_\mathcal{T}$, we describe what it does, and it will be apparent that is indeed expressible in MSOL.

5.1 Atom Encoding

One may think that, for a set $\mathcal{T} \in \mathbb{L}$, our MSOL sentence $\Phi_\mathcal{T}$ could directly talk about a chase path for $\mathcal{T}$. But this is not going to work for the simple reason that a chase path for $\mathcal{T}$ consists of infinitely many atoms. We therefore need something similar to a chase path, i.e., a structure that encodes a chase path for $\mathcal{T}$ as a labeled path, but much more parsimonious with respect to the labeling function. To this end, we need a convenient encoding for atoms.

We proceed to define a finite alphabet $\Lambda_\mathcal{T}$ that provides such an encoding. We write $\mathsf{EQ}_\mathcal{T}$ for the set of all equivalence relations on $\{f, m\} \times \{1, 2, \ldots \mathsf{ar}(\mathcal{T})\}$. The desired alphabet is defined as the set of triples

$$\Lambda_\mathcal{T} = \mathsf{sch}(\mathcal{T}) \times (\{\epsilon\} \cup \mathcal{T}) \times \mathsf{EQ}_\mathcal{T}.$$

Here is the idea underlying this encoding:

- The first element of each triple is a predicate; it simply tells us the predicate of the encoded atom.
- Concerning the second element, ϵ indicates that the encoded atom is the starting atom of the chase path. If an atom is an intermediate one, then the second element of the triple tells us from which TGD of $\mathcal{T}$ was generated.
- Finally, for the third element, note that f and m stand for "father" and "me". The idea is that, for example, the pair $((m, i), (m, j))$ says that the encoded atom has the same term at its i-th and j-th position, while the pair $((m, i), (f, j))$ says that the term at the i-th position in the atom in question is the same as the term at the j-th position of its father in the chase path.

For brevity, given a triple $\tau = (x, y, z) \in \Lambda_\mathcal{T}$, we write $\mathsf{pr}(\tau)$ for the predicate x, $\mathsf{org}(\tau)$ for y, i.e., the origin of the encoded atom, and $\mathsf{eq}(\tau)$ for the equivalence relation z.

5.2 Abstract Chase Paths

It should be intuitively clear that there exists a chase path for $\mathcal{T}$ iff the infinite path $v_0, v_1, \ldots$ can be labeled with triples from $\Lambda_\mathcal{T}$ in such a way that:

1. The label of v_0 is of the form (x, ϵ, z) for some predicate $x \in \mathsf{sch}(\mathcal{T})$, and an equivalence relation $z \in \mathsf{EQ}_\mathcal{T}$.
2. For each $i > 0$, it holds that
   (a) the atom encoded by the label of v_i can be obtained via an application of a trigger for $\mathcal{T}$ on the atom encoded by the label of v_{i-1}, and

[2] König's Lemma is a well-known result from graph theory, which states that an infinite rooted tree with finite out-degree has an infinite directed simple path starting from the root.

(b) there is no $j < i$ such that the atom encoded by the label of v_j stops the atom encoded by the label of v_i.

Such a $\Lambda_{\mathcal{T}}$-labeled infinite path is a structure that encodes a chase path for $\mathcal{T}$ using finitely many labels, which we call abstract chase path, for which our MSOL sentence could talk about. We proceed to formalize the above discussion.

We first need to make precise when a triple τ' is a successor of some triple τ, which means that the atom encoded by τ' can be obtained via an application of a trigger for $\mathcal{T}$ on the atom encoded by the label of τ. Consider a triple $\tau \in \Lambda_{\mathcal{T}}$. A *sucessor* of τ is a triple $\tau' \in \Lambda_{\mathcal{T}}$, with $\mathsf{org}(\tau') = \sigma$ for some $\sigma \in \mathcal{T}$, such that the following hold; for brevity, we write α for $\mathsf{body}(\sigma)$ and β for $\mathsf{head}(\sigma)$):

- $\mathsf{pr}(\tau)$ is the predicate of α.
- $\mathsf{pr}(\tau')$ is the predicate of β.
- $((m,i),(m,j)) \in \mathsf{eq}(\tau)$ iff $((f,i),(f,j)) \in \mathsf{eq}(\tau')$.
- $\alpha[i] = \beta[j]$ implies $((f,i),(m,j)) \in \mathsf{eq}(\tau')$.
- $\alpha[i] = \alpha[j]$ implies $((f,i),(f,j)) \in \mathsf{eq}(\tau')$.
- If $\beta[j]$ is an existentially quantified variable of σ, then $((m,i),(m,j)) \in \mathsf{eq}(\tau')$ iff $\beta[i] = \beta[j]$.

We also need to formalize when a triple τ stops a triple τ', which essentially means that the atom encoded by τ stops the atom encoded by τ'. This relies on the notion of correct coloring for a pair of triples. Consider two triples $\tau, \tau' \in \Lambda_{\mathcal{T}}$, with $\mathsf{pr}(\tau) = R/n$, $\mathsf{pr}(\tau') = R'/m$, and $\mathsf{org}(\tau') = \sigma$ for some $\sigma \in \mathcal{T}$. A pair of colors (i,j), where $i \in [n]$ and $j \in [m]$, is a *correct coloring* for (τ, τ') if σ is of the form

$$R(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n) \rightarrow \exists \bar{z}\, R'(y_1, \ldots, y_{j-1}, x, y_{j+1}, \ldots, y_m),$$

i.e., σ propagates the variable at position (R, i) in $\mathsf{body}(\sigma)$ to the position (R', j) in $\mathsf{head}(\sigma)$. Given a sequence of triples from $\Lambda_{\mathcal{T}}$ of the form $s = \tau_0, \ldots, \tau_n$, for $n > 0$, a *locally correct coloring* for s is a tuple of colors $(i_0, \ldots, i_n)$ such that, for each $0 \leq j < n$, (i_j, i_{j+1}) is a correct coloring for (τ_j, τ_{j+1}).

Assume now that $\mathsf{org}(\tau_n) = \sigma$ for some TGD $\sigma \in \mathcal{T}$. We say that τ_0 *stops* τ_n *w.r.t.* s, written $\tau_0 \blacktriangleright_s \tau_n$, if:

- $\mathsf{pr}(\tau_0) = \mathsf{pr}(\tau_n)$, and $((m,i),(m,j)) \in \mathsf{eq}(\tau_n)$ implies $((m,i),(m,j)) \in \mathsf{eq}(\tau_0)$.
- If $\mathsf{head}(\sigma)[j] \in \mathsf{fr}(\sigma)$, then there exists a locally correct coloring $(i, \ldots, j)$ for s, and $((m,i),(m,j)) \in \mathsf{eq}(\tau_0)$.

The first condition ensures that there exists a homomorphism h from the atom encoded by τ_n to the atom encoded by τ_0, while the second condition ensures that h is the identity on the witnesses for the variables in $\mathsf{fr}(\sigma)$.

We now have all the ingredients needed to formally define the notion of abstract chase path:

Definition 3 (Abstract Chase Path) Let $\mathcal{T} \in \mathbb{L}$. An *abstract chase path for* $\mathcal{T}$ is an infinite $\Lambda_{\mathcal{T}}$-labeled path $(v_i)_{i \geq 0}$, with λ being the labeling function, such that:

1. $\mathsf{org}(v_0) = \epsilon$.
2. For each $i > 0$, it holds that
   (a) $\lambda(v_i)$ is a successor of $\lambda(v_{i-1})$, and
   (b) there is no $0 \leq j < i$ such that $\lambda(v_j) \blacktriangleright_s \lambda(v_i)$ with $s = \lambda(v_j), \lambda(v_{j+1}), \ldots, \lambda(v_i)$.

□

It should not be difficult to observe the correspondence between the two conditions in the definition of chase path (Definition 2) and the two conditions in the definition of abstract chase path. This allows us to show the following:

Lemma 2 *Let $\mathcal{T} \in \mathbb{L}$. The following are equivalent*:

1. *There exists a chase path for $\mathcal{T}$.*
2. *There exists an abstract chase path for $\mathcal{T}$.*

Proof The proof relies on the obvious translation of a chase path for $\mathcal{T}$ into an abstract chase path for $\mathcal{T}$, and vice-versa. We proceed to make this explicit.

(1) $\Rightarrow$ (2). By hypothesis, there exists a chase path $(\alpha_i)_{i \geq 0}$ for $\mathcal{T}$. By definition, for each $i > 0$, there exists a trigger (σ_i, h_i) for $\mathcal{T}$ on $\{\alpha_{i-1}\}$ such that $\alpha_i = \mathsf{result}(\sigma_i, h_i)$, and there is no $0 \leq j < i$ such that $\alpha_j \blacktriangleright \alpha_i$. We proceed to construct an infinite $\Lambda_{\mathcal{T}}$-labeled path $p = (v_i)_{i \geq 0}$ as follows:

- The node v_0 is labeled with (x, ϵ, z), where x is the predicate of α_0, and z is the smallest equivalence relation on $\{f, m\} \times \{1, \ldots, \mathsf{ar}(\mathcal{T})\}$ such that, for every $1 \leq i < j \leq \mathsf{ar}(x)$, $\alpha_0[i] = \alpha[j]$ implies $((m,i),(m,j)) \in z$.
- For each $i > 0$, v_i is labeled with (x, σ_i, z), where x is the predicate of α_i, and z is the smallest equivalence relation on $\{f, m\} \times \{1, \ldots, \mathsf{ar}(\mathcal{T})\}$ such that the following hold; we assume that P is the predicate of α_{i-1}:
  (i) for each $1 \leq j < k \leq \mathsf{ar}(P)$, $\alpha_{i-1}[j] = \alpha_{i-1}[k]$ implies $((f,j),(f,k)) \in z$,
  (ii) for each $1 \leq j \leq \mathsf{ar}(P)$ and $1 \leq k \leq \mathsf{ar}(x)$, $\alpha_{i-1}[j] = \alpha_i[k]$ implies $((f,j),(m,k)) \in z$, and
  (iii) for each $1 \leq j < k \leq \mathsf{ar}(x)$, $\alpha_i[j] = \alpha_i[k]$ implies $((m,j),(m,k)) \in z$.

This completes the construction of p. It is now not difficult to see that p is an abstract chase path for $\mathcal{T}$. Indeed, conditions 1 and 2a of Definition 3 can be easily verified. Concerning condition 2b, by contradiction, assume there exists $i > 0$ and $0 \leq j < i$ such that (with λ being the labeling function of p) $\lambda(v_j) \blacktriangleright_s \lambda(v_i)$ with $s = \lambda(v_j), \lambda(v_{j+1}), \ldots, \lambda(v_i)$. But this implies that $\alpha_j \blacktriangleright \alpha_i$, which is a contradiction.

(2) $\Rightarrow$ (1). By hypothesis, there exists an abstract chase path $p = (v_i)_{i \geq 0}$ for $\mathcal{T}$; λ is the labeling function of p. By definition, for each $i > 0$, $\lambda(v_i)$ is a successor of $\lambda(v_{i-1})$, and there is no $0 \leq j < i$ such that $\lambda(v_j) \blacktriangleright_s \lambda(v_i)$ with $s = \lambda(v_j), \lambda(v_{j+1}), \ldots, \lambda(v_i)$.

We inductively construct an infinite sequence $(\alpha_i)_{i \geq 0}$ of atoms over $\mathsf{sch}(\mathcal{T})$, which contain constants and nulls, as follows:

- The atom α_0 is of the form $P(c_1, \ldots, c_n)$, with $c_1, \ldots, c_n$ being constants, where $P = \mathsf{pr}(\lambda(v_0))$, and, for each $i, j \in [n]$, $c_i = c_j$ iff $((m, i), (m, j)) \in \mathsf{eq}(\lambda(v_0))$.
- Assume that we have constructed $\alpha_0, \alpha_1, \ldots, \alpha_i$; let R being the predicate of α_i. We define α_{i+1} as an atom of the form $P(t_1, \ldots, t_n)$, with $t_1, \ldots, t_n$ being constants and nulls, where $P = \mathsf{pr}(\lambda(v_{i+1}))$, and
  - (i) for each $j, k \in [n]$, $t_j = t_k$ iff $((m, j), (m, k)) \in \mathsf{eq}(\lambda(v_{i+1}))$,
  - (ii) for each $1 \leq j \leq \mathsf{ar}(R)$ and $1 \leq k \leq \mathsf{ar}(P)$, if $((f, j), (m, k)) \in \mathsf{eq}(\lambda(v_{i+1}))$, then $\alpha_i[j] = \alpha_{i+1}[k]$,
  - (iii) for each $1 \leq j \leq \mathsf{ar}(P)$, with $\sigma = \mathsf{org}(\lambda(v_{i+1}))$, if $\mathsf{head}(\sigma)[j]$ is an existentially quantified variable z, then $\alpha_{i+1}[j]$ is the null $\perp^z_{\sigma,h}$ where h is the homomorphism from $\mathsf{body}(\sigma)$ to α_i, which exists since $\lambda(v_{i+1})$ is a successor of $\lambda(v_i)$.

This completes the construction of $(\alpha_i)_{i \geq 0}$. It is not difficult to see that $(\alpha_i)_{i \geq 0}$ is a chase sequence for $\mathcal{T}$. Indeed, conditions 1 and 2a of Definition 2 can be easily verified. Concerning condition 2b, by contradiction, assume that there exists $i > 0$ and $0 \leq j < i$ such that $\alpha_j \blacktriangleright \alpha_i$. But this implies that $\lambda(v_j) \blacktriangleright_s \lambda(v_i)$ with $s = \lambda(v_j), \lambda(v_{j+1}), \ldots, \lambda(v_i)$, which is a contradiction, and the claim follows. □

5.3 Abstract Chase Paths are MSOL-definable

Our last task is to show the following:

Lemma 3 *Let $\mathcal{T} \in \mathbb{L}$. There is an MSOL sentence $\Phi_{\mathcal{T}}$ such that, for an infinite $\Lambda_{\mathcal{T}}$-labeled path p, it holds that $p \models \Phi_{\mathcal{T}}$ iff p is an abstract chase path for $\mathcal{T}$.*

The sentence $\Phi_{\mathcal{T}}$ has to check whether an infinite $\Lambda_{\mathcal{T}}$ -labeled path $p = (V, <)$, with λ being the labeling function, enjoys the two conditions of Definition 3. Let us assume, for the moment, that we have available the following formulas:

- $\phi_{\text{desc}}(x, y)$ states that x is a descendant of y.
- $\phi_{\text{stop}}(x, y)$ means that $\lambda(x) \blacktriangleright_s \lambda(y)$ with s being the sequence of triples that label the subpath from x to y.

By exploiting the above formulas, we can devise $\Phi_{\mathcal{T}}$ as the conjunction $\psi_1 \wedge \psi_{2a} \wedge \psi_{2b}$, where each conjunct checks for the corresponding condition of Definition 3. Let

$$\mathsf{fact} = \{(x, y, z) \in \Lambda_{\mathcal{T}} \mid y = \epsilon\}$$

i.e., the set of triples of $\Lambda_{\mathcal{T}}$ that correspond to facts, and

$$\mathsf{succ} = \{(\tau, \tau') \in \Lambda_{\mathcal{T}} \times \Lambda_{\mathcal{T}} \mid \tau' \text{ is a successor of } \tau\}.$$

We also use unary predicates of the form L_τ, where $\tau \in \Lambda_{\mathcal{T}}$, to state that the label of a node should be τ. The sentences ψ_1, ψ_{2a} and ψ_{2b} are defined as follows:

$$\psi_1 = \forall x \left(\left((\neg \exists y\, (y < x)) \rightarrow \bigvee_{\tau \in \mathsf{fact}} L_\tau(x) \right) \right)$$

$$\psi_{2a} = \forall x \forall y \left((x < y) \rightarrow \bigvee_{(\tau, \tau') \in \mathsf{succ}} (L_\tau(x) \wedge L_{\tau'}(y)) \right)$$

and

$$\psi_{2b} = \forall x \left(\neg \exists y \left(\phi_{\text{desc}}(y, x) \wedge \phi_{\text{stop}}(y, x) \right) \right).$$

We proceed to give more details about the auxiliary formulas used in $\Phi_{\mathcal{T}}$. The formula $\phi_{\text{desc}}(x, y)$ comes from the general MSOL toolbox. It simply states that any set of nodes that contains x and is closed under $<$, it also contains y.

The formula $\phi_s(x, y)$ can be easily devised providing that we have available a formula $\phi^{i,j}_{=}(x, y)$, for each $i, j \in [\mathsf{ar}(\mathcal{T})]$, that states the following: the term in the atom encoded by the label of x at position i is equal to the term in the atom encoded by the label of y at position j. This can be expressed in MSOL as follows: there exists a subset A of the nodes in the $\Lambda_{\mathcal{T}}$-labeled path p such that:

1. A is the path with x and y being its ends, i.e., A is finite, x, y have exactly one neighbor in A, and any other node in A has exactly two neighbors, and
2. A is a disjoint union of the sets $A_1, \ldots, A_{\mathsf{ar}(\mathcal{T})}$ such that $x \in A_i$, $y \in A_j$, and, for all pairs $z, w \in A$ with (z, w) being an edge in the path p, $z \in A_k$ and $w \in A_\ell$, it holds that $((f, k), (m, \ell)) \in \mathsf{eq}(\lambda(w))$.

Note that above we need to check whether the set of nodes A is finite. This can be done by exploiting a formula that comes from the general MSOL toolbox. It simply states that p has

an infinite directed subpath, starting from some intermediate node p, which is disjoint with A.

We are now ready to conclude the proof of Theorem 2. By Lemmas 1, 2 and 3, the following are equivalent:

1. $\mathcal{T} \notin \mathbb{CT}^{\mathsf{res}}_{\forall\forall}$.
2. $\Phi_{\mathcal{T}}$ is satisfiable over infinite $\Lambda_{\mathcal{T}}$-labeled paths.

Since the latter is decidable, Theorem 2 follows.

6 Conclusions

We presented an alternative proof via standard means for the decidability of all-instances restricted chase termination in the case of single-head linear TGDs. Our proof exploits MSOL, and we believe that is simpler than the ones obtained from the literature. Despite the simplicity of linear TGDs, there are still a couple of challenging questions concerning all-instances restricted chase termination that remain open:

1. What about the exact complexity of the problem?
2. What about arbitrary linear TGDs that can have a conjunction of atoms in the head?

Acknowledgements We thank the referees for their useful feedback. Gogacz was supported by Poland's National Science Centre grant 2018/30/E/ST6/00042. Marcinkowski was supported by the NCN grant 2016/23/B/ST6/01438. Pieris was supported by the EPSRC grant EP/S003800/1 "EQUID".

References

1. Aho Alfred V, Sagiv Yehoshua, Ullman Jeffrey D (1979) Efficient optimization of a class of relational expressions. ACM Trans. Database Syst 4(4):435–454
2. Baget J-F, Leclère M, Mugnier M-L, Salvat E (2011) On rules with existential variables: walking the decidability line. Artif Intell 175(9–10):1620–1654
3. Baget J-F, Mugnier M-L, Rudolph S, Thomazo M (2011) Walking the complexity lines for generalized guarded existential rules. In: IJCAI, 712–717
4. Bednarczyk B, Ferens R, Ostropolski-Nalewaja P (2020) All-instances oblivious chase termination is undecidable for single-head binary tgds. In: IJCAI, 1719–1725
5. Beeri C, Vardi MY (1984) A proof procedure for data dependencies. J ACM 31(4):718–741
6. Benedikt M, Konstantinidis G, Mecca G, Motik B, Papotti P, Santoro D, Tsamoura E (2017) Benchmarking the chase. In: PODS, pp 37–52
7. Calautti M, Gottlob G, Pieris A (2015) Chase termination for guarded existential rules. In: PODS, pp 91–103
8. Calautti M, Pieris A (2019) Oblivious chase termination: the sticky case. In: ICDT pp 17:1–17:18
9. Calì A, Gottlob G, Kifer M (2013) Taming the infinite chase: query answering under expressive relational constraints. J Artif Intell Res 48:115–174
10. Calì A, Gottlob G, Lukasiewicz T (2012) A general Datalog-based framework for tractable query answering over ontologies. J Web Sem 14:57–83
11. Calì Andrea, Gottlob Georg, Pieris Andreas (2012) Towards more expressive ontology languages: the query answering problem. Artif Intell 193:87–128
12. Deutsch A, Nash A, Remmel JB (2008) The chase revisisted. In: PODS, pp 149–158
13. Deutsch A, Tannen V (2003) Reformulation of XML queries and constraints. In: ICDT, pp 225–241
14. Fagin R, Kolaitis PG, Miller RJ, Popa L (2005) Data exchange: semantics and query answering. Theor Comput Sci 336(1):89–124
15. Gogacz T, Marcinkowski J (2014) All-instances termination of chase is undecidable. In: ICALP, pp 293–304
16. Gogacz T, Marcinkowski J, Pieris A (2020) All-Instances restricted chase termination. In: PODS, pp 245–258
17. Grau BC, Horrocks I, Krötzsch M, Kupke C, Magka D, Motik B, Wang Z (2013) Acyclicity notions for existential rules and their application to query answering in ontologies. J Artif Intell Res 47:741–808
18. Greco S, Spezzano F, Trubitsyna I (2011) Stratification criteria and rewriting techniques for checking chase termination. PVLDB 4(11):1158–1168
19. Hernich A, Schweikardt N (2007) Cwa-solutions for data exchange settings with target dependencies. In: PODS, pp 113–122
20. Krötzsch M, Marx M, Rudolph S (2009) The power of the terminating chase (invited talk). In: ICDT, pp 3:1–3:17
21. Leclère M, Mugnier M-L, Thomazo M, Ulliana F (2019) A single approach to decide chase termination on linear existential rules. In: ICDT, pp 18:1–18:19
22. Maier D, Mendelzon AO, Sagiv Y (1979) Testing implications of data dependencies. ACM Trans Database Syst 4(4):455–469
23. Marnette B (2009) Generalized schema-mappings: from termination to tractability. In: PODS, pp 13–22
24. Meier M, Schmidt M, Lausen G (2009) On chase termination beyond stratification. PVLDB 2(1):970–981
25. Nenov Y, Piro R, Motik B, Horrocks I, Wu Z, Banerjee J (2015) Rdfox: a highly-scalable rdf store. In: ISWC, pp 3–20
26. Urbani J, Krötzsch M, Jacobs CJH, Dragoste I, Carral D (2018) Efficient model construction for horn logic with VLog—system description. In: IJCAR, pp 680–688